01/10/2023

Διαδικτυακή Εφαρμογή Ενοικίασης Δωματίων

Τεχνολογίες Εφαρμογών Διαδικτύου

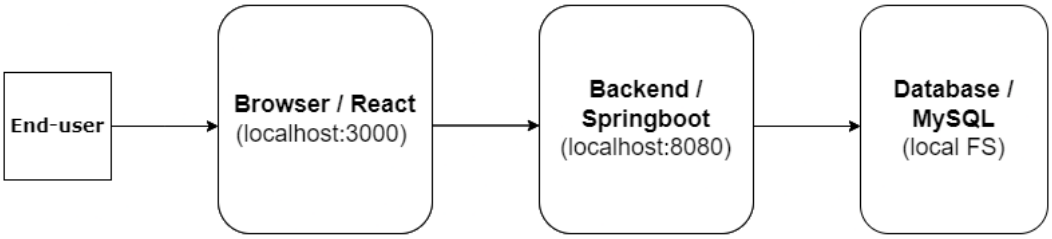
Κωνσταντίνος Μπιμπής: 1115201900125
Άγγελος Σκέρτσος: 1115201900172

Πίνακας Περιεχομένων

Κεφάλαιο 1 Εισαγωγή

Σκοπός της ακόλουθης εργασίας ήταν η δημιουργία μιας σύγχρονης διαδραστικής διαδικτυακής εφαρμογής με αντικείμενο την ενοικίαση δωματίων, με μοντέλο παρόμοιο της AirBnB. Ένας χρήστης έχει έναν ή παραπάνω από τους εξής τρεις ρόλους: διαχειριστής (**admin**), οικοδεσπότης (**host**), και ενοικιαστής (**tenant**). Ένας οικοδεσπότης καταθέτει ένα δωμάτιο (**rental**) προς ενοικίαση, το οποίο μπορεί να δει ύστερα ένας ενοικιαστής, να κάνει κράτηση(**booking**), και να ανεβάσει μια κριτική (**review**). Ύστερα, μπορεί να αναζητήσει προτεινόμενα δωμάτια με βάση το ιστορικό του. Ο διαχειριστής μπορεί να δει τη λίστα με όλους τους χρήστες, να επικυρώσει πιθανούς οικοδεσπότες, και να αποθηκεύσει ένα στιγμιότυπο της βάσης δεδομένων.

Η διεπαφή με τον χρήση γίνεται αποκλειστικά στον browser, με τοπική σύνδεση(localhost). Την αναλαμβάνει ένας εμπρόσθιος(frontend) server, ο οποίος μεταφέρει και αντλεί τις κατάλληλες πληροφορίες από τον κεντρικό(backend) server, ο οποίος διαχειρίζεται την επικοινωνία με την τοπικά αποθηκευμένη βάση δεδομένων.



Η εφαρμογή χρησιμοποιεί **RESTful api** για τη μεταβίβαση πληροφορίας μεταξύ των servers και δεν χρησιμοποιούνται ***sessions*** για την επαλήθευση ταυτότητας. Η εγγύηση ασφάλειας και η επαλήθευση ταυτότητας επιτυγχάνεται με τη χρήση **Json Web Tokens(jwt)**. Ένα jwt δημιουγείται από τον κεντρικό server όταν ένας χρήστης συνδέεται (login) στην εφαρμογή και πληροφορίες για τον χρήστη, συμπεριλαμβανομένων και των ρόλων του. Επικοινωνείται ύστερα στον browser, και χρησιμοποιείται στην επικεφαλίδα κάθε αίτησης στον server (εφ' όσον χρειάζεται) για την επαλήθευση της ταυτότητας και των ρόλων του χρήστη στην εφαρμογή.

Όταν ένας χρήστης πλοηγείται στην εφαρμογή, προτρέπεται να συνδεθεί / εγγραφεί, ή να μεταφερθεί στην αρχική σελίδα της εφαρμογής ως ανώνυμος χρήστης.

Στην αρχική σελίδα, μπορεί να διαλέξει χώρα, πόλη και γειτονιά, καθώς και αριθμό ατόμων και ημερομηνίες. Θα του εμφανιστούν όλα τα δωμάτια που πληρούν τις προϋποθέσεις (να είναι διαθέσιμα εκείνες τις μέρες, να μπορούν να υποστηρίξουν αριθμό ατόμων, κλπ), σελιδοποιημένα σε σελίδες των 10 και ταξινομημένα σε αύξουσα τιμή. Ύστερα μπορεί να περιορίσει περαιτέρω τον αριθμό των δωματίων επιλέγοντας τύπο δωματίου, μέγιστη τιμή, και διάφορες παροχές (A/C, πάρκινγκ κλπ). Μπορεί να μεταφερθεί σε προηγούμενη/επόμενη ή πρώτη/τελευταία σελίδα με τη χρήση κουμπιών. Πατώντας ένα δωμάτιο, μεταφέρεται στη σελίδα με τις πληροφορίες του. Αν είναι συνδεδεμένος και έχει τον ρόλο του ενοικιαστή, μπορεί επίσης από τη σελίδα του δωματίου να κάνει κράτηση, κριτική (αν το έχει κλείσει στο παρελθόν) ή να μεταφερθεί σε σελίδα ανταλλαγής μηνυμάτων με τον οικοδεσπότη.

Ένας οικοδεσπότης μπορεί να προσθέσει ένα δωμάτιο προς ενοικίαση ή να δει μια λίστα με τα υπάρχοντα δωμάτιά του. Διαλέγοντας ένα δωμάτιο από τη λίστα, μπορεί να ανανεώσει τις πληροφορίες του (να προσθέσει / αφαιρέσει φωτογραφίες, να αλλάξει τιμή ανά άτομο κλπ) ή να μεταφερθεί στη σελίδα μηνυμάτων για το συγκεκριμένο δωμάτιο. Εκεί του

εμφανίζεται η λίστα πιθανών ενοικιαστών, ταξινομημένη με βάση το τελευταίο μήνυμα του καθενός. Πατώντας το αντίστοιχο κουμπί πλοηγείται στη σελίδα μηνυμάτων με τον αντίστοιχο επικείμενο ενοικιαστή.

Ενοικιαστές και οικοδεσπότες μπορούν να πλοηγηθούν στη σελίδα του προφίλ τους, όπου μπορούν να ανανεώσουν τα στοιχεία τους (αλλαγή φωτογραφίας προφίλ, e-mail κλπ).

1.1 Εργαλεία που χρησιμοποιήθηκαν:

Λειτουργία	**Όνομα**	**Έκδοση**
Βάση Δεδομένων	MySQL	8
Επικοινωνία Backend – Βάση Δεδομένων	Hibernate	6.2.6
Backend	SpringBoot	3.1.2
Frontend	React	9.8.1
Σύνταξη(compilation) του project	Maven	-

Κεφάλαιο 2 Οδηγίες Εγκατάστασης και Εκτέλεσης

2.1 Εγκατάσταση

Έστω **{project_folder}** το absolute path του top-level φακέλου του project.

2.1.1 Frontend

Για την εκτέλεση του frontend είναι απαραίτητη η εγκατάσταση του NodeJs και του node-package-manager(npm) από το επίσημο site. Ύστερα, πρέπει να κληθούν οι παρακάτω εντολές στο {project_folder}/web:

- npm install

Για εξοικονόμηση χώρου, στο .zip αρχείο δεν περιέχεται ο φάκελος {project_folder}/web/node_modules. Συνεπώς είναι απαραίτητο να εκτελεστεί αυτή η εντολή για την αυτόματη δημιουργία του

- npm install react
- npm install react-datepicker
- npm install react-dom
- npm install react-multi-date-picker
- npm install react-router-dom
- npm install react-scripts

2.1.2 Backend

Το backend έχει γραφτεί σε **Java 17**, άρα είναι απαραίτητη η χρήση αντίστοιχου compiler

Εγκατάσταση maven:

Το project χρησιμοποιεί maven για το compilation, οπότε είναι απαραίτητη η εγκατάστασή του. Σε windows powershell, στον {project_folder} πρέπει να εκτελεστεί η εντολή
.\apache-maven-3.9.4\bin\mvn install

2.2 Certificates

Η εφαρμογή χρησιμοποιεί το πρωτόκολλο **SSL/TLS** , και συγκεκριμένα **self-signed certificate** για την κρυπτογράφηση των HTTP αιτημάτων. Αυτό απαιτεί εγκατάσταση τόσο στο backend όσο στο frontend

2.2.1 Frontend

Εγκατάσταση του **mkcert**:
Σε ένα elevated windows powershell πρέπει να εκτελεστεί η εντολή :
choco install mkcert

Δημιουργία frontend certificate :
Όταν έχει γίνει εγκατάσταση του mkcert, σε ένα powershell στον φάκελο {project_folder}/web/react_certificates να εκτελεστεί η εντολή :
mkcert -install localhost

2.2.2 Backend

Εγκατάσταση του **openssl** :
Σε ένα elevated windows powershell να εκτελεστεί η εντολή :
choco install openssl

Δημιουργία backend certificate :
Όταν έχει γίνει εγκατάσταση του openssl, σε ένα elevated windows powershell να εκτελεστεί η εντολή :

openssl pkcs12 -export -out
{absolute_path}\src\main\resources\certificates\backendCertificate.p12 -inkey
{absolute_path}\web\react_certificates\localhost-key.pem -in
{absolute_path}\web\react_certificates\localhost.pem

Μετά απ' την εκτέλεση της παραπάνω εντολής, θα ζητηθεί είσοδος κάποιου passwordτο οποίο πρέπει να είναι : **password**

Όταν γίνουν τα παραπάνω, θα πρέπει στον φάκελο {absolute_path}\src\main\resources\certificates να υπάρχει το αρχείο **backendCertificate.p12** και στον φάκελο {absolute_path}\web\react_certificates, να υπάρχουν τα αρχεία **localhost-key.pem** και **localhost.pem**

2.3 Εκτέλεση εφαρμογής

Για τη χρήση της εφαρμογής, πρέπει να είναι ενεργοί ταυτόχρονα και οι 2 servers, frontend και backend, σε διαφορετικά terminals.

2.3.1 Frontend

Σε ένα οποιοδήποτε shell, στον φάκελο {project_folder}/web πρέπει να εκτελεστεί η εντολή
npm start

2.3.2 Backend

Σε ένα windows powershell, στον φάκελο {project_folder} να εκτελεστεί η εντολή:
java -jar .\target\AirBnBClone-0.0.1-SNAPSHOT.jar

Κεφάλαιο 3 Οντότητες – Γενική Επισκόπηση

Σε αυτό το κεφάλαιο θα γίνει μια γενική επεξήγηση των σημαντικότερων οντοτήτων στη βάση δεδομένων και του ρόλου τους στην εφαρμογή.

3.1 Χρήστης (User)

Ένας **χρήστης (user)** δημιουργείται κατά την εγγραφή (register) στην εφαρμογή. Περιέχει πληροφορίες όπως το ονοματεπώνυμο, username, password κλπ. Όλα τα στοιχεία του είναι μεταβλητά μέσω μιας σελίδας επεξεργασίας προφίλ, εκτός από τους **ρόλους (roles)** του στην εφαρμογή (μεταξύ των admin, host, tenant), οι οποίοι επιλέγονται από τον επικείμενο χρήστη κατά την εγγραφή στην εφαρμογή και δεν είναι δυνατόν να αλλάξουν.

3.2 Ρόλος (Role)

Οι **ρόλοι(roles)** ενός χρήστη καθορίζουν όχι μόνο τις δυνατότητές του, αλλά και τη γραφική διεπαφή της εφαρμογής. Ένας χρήστης που έχει *και* τον ρόλο του οικοδεσπότη, *και* του ενοικιαστή μπορεί να διαλέξει οποιαδήποτε στιγμή ποια από τις δύο διεπαφές θέλει να χρησιμοποιήσει.

Η **διεπαφή του ενοικιαστή** περιλαμβάνει αρχικά το πάνελ αναζήτησης δωματίων και ένα κουμπί εμφάνισης προτεινόμενων δωματίων με βάση το ιστορικό του. Και οι δύο επιλογές εμφανίζουν μια λίστα από δωμάτια, με μερικές βασικές πληροφορίες για κάθε δωμάτιο όπως τίτλο, φωτογραφία, κριτικές κ.ο.κ. Η αναζήτηση να προσφέρει επιπρόσθετα την επιλογή φιλτραρίσματος με βάση την ύπαρξη παροχών όπως πάρκινγκ, ασανσέρ κ.ο.κ. Πατώντας ένα δωμάτιο, μεταφέρεται σε σελίδα με αναλυτικές πληροφορίες για το δωμάτιο, καθώς και βασικές πληροφορίες για τον οικοδεσπότη. Έχει την επιλογή να κάνει κράτηση για τις ημερομηνίες και τα άτομα που είχε επιλέξει, να προσθέσει μια κριτική (αν έχει κάνει κράτηση στο παρελθόν), ή να πλοηγηθεί σε σελίδα ανταλλαγής μηνυμάτων με τον οικοδεσπότη πατώντας το πάνελ με τις πληροφορίες του.

Η **διεπαφή του οικοδεσπότη** είναι διαθέσιμη μόνο αν έχει επικυρωθεί από τον διαχειριστή. Διαφορετικά, εμφανίζεται σχετικό μήνυμα και απαγορεύεται η χρήση της διεπαφής του οικοδεσπότη. Όταν επικυρωθεί ο χρήστης, του δίνονται οι επιλογές να προσθέσει καινούριο δωμάτιο ή να δει μια λίστα με τα δικά του. Για να προσθέσει καινούριο δωμάτιο, αρκεί να συμπληρώσει τα υποχρεωτικά πεδία με τις πληροφορίες του και να επιβεβαιώσει τις επιλογές του. Στη λίστα με τα δωμάτιά του, πατώντας ένα δωμάτιο πλοηγείται στη σελίδα με τις αναλυτικές του λεπτομέρειες, από όπου μπορεί να ανανεώσει όλες τις πληροφορίες του, επιβεβαιώνοντάς το με το πάτημα ενός κουμπιού.

Και οι δύο διεπαφές δίνουν στον χρήστη την επιλογή να ανανεώσει τα στοιχεία του προφίλ του με σχετικό κουμπί, καθώς και την επιλογή να μεταφερθούν στην άλλη διεπαφή, εφ’όσον ο χρήστης έχει τον αντίστοιχο ρόλο.

Η **διεπαφή του διαχειριστή** τού δίνει τη δυνατότητα να δει τη λίστα όλων των χρηστών, και να κατεβάσει ένα στιγμιότυπο των πιο σημαντικών στοιχείων της βάσης δεδομένων σε μορφή .json ή .xml. Από τη λίστα χρηστών, πατώντας σε έναν χρήστη πλοηγείται σε σελίδα με τις αναλυτικές πληροφορίες του. Από εκεί μπορεί να επικυρώσει την ιδιότητα ενός επικείμενου οικοδεσπότη.
Πατώντας κουμπί για λήψη της βάσης δεδομένων, του επιστρέφεται ένα αρχείο .json ή .xml αντίστοιχα, το οποίο μπορεί να αποθηκεύσει τοπικά στον υπολογιστή του.
Δημιουργείται ένα στιγμιότυπο για καθέναν από τους τρεις ρόλους κατά την εκκίνηση του κεντρικού server, το οποίο ενσωματώνεται κατά τη δημιουργία ενός νέου χρήστη στους ρόλους του, εφόσον έχει επιλεχθεί.

3.3 Φωτογραφία (Photo)

Μια **φωτογραφία** ορίζεται από αναγνωριστικό της (**id**), το **όνομά (name)** της, τον **τύπο (contentType)** της (.jpg, .png), και το σχετικό μονοπάτι (**filePath**) της στον τοπικά αποθηκευμένο φάκελο της βάσης δεδομένων, αλλά η ίδια η φωτογραφία δεν αποθηκεύεται στη βάση για την εξοικονόμηση χώρου. Όταν ένας χρήστης ανεβάζει μια φωτογραφία στη βάση δεδομένων (φωτογραφία προφίλ / δωματίου), στέλνεται ολόκληρη στον κεντρικό server και αντιγράφεται τοπικά, σε αντίστοιχο φάκελο χρήστη / δωματίου, αλλά στη βάση αποθηκεύονται μόνο τα 4 προαναφερθέντα πεδία της. Η εμφάνιση της φωτογραφίας γίνεται μέσω του <img> tag της html, με βάση το μονοπάτι στο οποίο είναι αποθηκευμένη. Όταν ένας χρήστης ανανεώνει υπάρχουσα φωτογραφία προφίλ ή ένας ενοικιαστής διαγράφει φωτογραφίες από ένα δωμάτιό του, οι σχετικές φωτογραφίες διαγράφονται από τη βάση δεδομένων.

3.4 Ενοικιαζόμενο (rental)

Ένα **ενοικιαζόμενο (rental)** αντιπροσωπεύει ένα δωμάτιο / σπίτι που έχει τεθεί από έναν οικοδεσπότη για κράτηση / ενοικίαση από ενοικιαστές. Περιλαμβάνει (μεταξύ άλλων) **τίτλο**, **οικοδεσπότη(host),** λίστα με **διαθέσιμες ημερομηνίες(availableDates)** στις οποίες μπορεί να γίνει κράτηση, **διεύθυνση(address), μέγιστο αριθμό καλεσμένων(maxGuests)** για κάθε κράτηση, **τύπο(type)** ανάμεσα δημοσίου δωματίου(publicRoom), ιδιωτικού δωματίου(privateRoom) ή σπιτιού(house), μια λίστα από **κριτικές(reviews)**, και διάφορα πεδία σχετικά με λεπτομέρειες όπως αριθμός κρεβατιών κλπ, και κανόνες όπως αν επιτρέπεται το κάπνισμα. Τα πεδία που σχετίζονται με την τιμή κράτησης είναι η **βασική τιμή(basePrice)** (τιμή για 1 μέρα, με 0 άτομα) και **κόστος ανά άτομο(chargePerPerson)** (έξτρα κόστος ανά άτομο, ανά ημέρα ενοικίασης). Η τελική τιμή ενοικίασης ορίζεται απ' τον τύπο

Τιμή Ενοικίασης =
(βασική_τιμή + αριθμός_ατόμων * κόστος_ανά_άτομο) * ημέρες_κράτησης

3.5 Κράτηση (Booking)

Μια **κράτηση (booking)** περιλαμβάνει (μεταξύ άλλων) τον **ενοικιαστή(booker)**, ο οποίος είναι ένας χρήστης με τον αντίστοιχο ρόλο, το **ενοικιαζόμενο(rental),** τις μέρες κράτησης που ορίζονται από την **πρώτη (startDate)** και την **τελευταία (endDate)**, τον **αριθμό των ατόμων (guests)**, την **τιμή κράτησης(price)** και τη **στιγμή δημιουργίας της κράτησης(bookedAt).** Για να γίνει μια κράτηση, πρέπει το ενοικιαζόμενο να είναι διαθέσιμο τις αντίστοιχες ημέρες, το πλήθος των ημερών να μην είναι μικρότερο από τις ελάχιστες μέρες κράτησης του ενοικιαζόμενου, και ο αριθμός των ατόμων να μην υπερβαίνει τον μέγιστο αριθμό ατόμων του ενοικιαζόμενου. Όταν

γίνεται μια κράτηση, οι μέρες κράτησης αφαιρούνται από τις διαθέσιμες ημέρες του ενοικιαζόμενου.

3.6 Κριτική (Review)

Όταν γίνεται μια **κριτική (review)** από έναν **ενοικιαστή (reviewer)** σε ένα **ενοικιαζόμενο (rental)**, αποθηκεύεται η **χρονική στιγμή της κριτικής (issuedAt)**, και περιέχει **κείμενο (text)**, και μια **βαθμολόγηση** από το 1 ως το 5 **(stars)**. Για να μπορέσει να κάνει κριτική, ο ενοικιαστής πρέπει να έχει κάνει κράτηση του ενοικιαζόμενου στο παρελθόν.

3.7 Μήνυμα – Ιστορικό Μηνυμάτων

Ένα **μήνυμα (message)** περιλαμβάνει τον **πομπό (sender)**, τη **χρονική στιγμή αποστολής (sentAt)**, και το **περιεχόμενο (contents)**. Ο πομπός είναι χρήστης ανεξαρτήτως ρόλου. Στέλνεται στο πλαίσιο της επικοινωνίας ενός ενοικιαστή με τον οικοδεσπότη ενός συγκεκριμένου ενοικιαζόμενου. Τα μηνύματα ομαδοποιούνται σε **ιστορικό μηνυμάτων (message history),** που καταγράφουν επίσης τον **ενοικιαστή (tenant)**, και το **ενοικιαζόμενο (rental)** για το οποίο γίνεται η συζήτηση. Συνεπώς, το ιστορικό περιλαμβάνει τα μηνύματα του ενοικιαστή και του οικοδεσπότη του ενοικιαζόμενου.

3.8 Αναζήτηση – Ιστορικό Αναζητήσεων

Μια **αναζήτηση (search)** περιλαμβάνει τη χώρα(country), πόλη(city), γειτονιά(neighbourhood), πρώτη και τελευταία μέρα κράτησης (startDate / endDate), τον αριθμό ατόμων(guests), και επιλογές σχετικά με τις διάφορες παροχές των ενοικιαζόμενων όπως wi-fi, κουζίνα κλπ, που επιλέχθηκαν για αναζήτηση από έναν ενοικιαστή. Ομαδοποιούνται σε **ιστορικό αναζήτησης (search history)** για κάποιον **χρήστη (user)**, μαζί με ένα **map** με **κλειδιά τα ενοικιαζόμενα** των οποίων τη σελίδα επισκέφθηκε, και **τιμές τον αριθμό των φορών** που επισκέφθηκε αυτές τις σελίδες. Οι δύο παραπάνω οντότητες (search, search history) χρησιμοποιούνται αποκλειστικά για τη δημιουργία εξατομικευμένης λίστας με προτεινόμενων ενοικιαζόμενων.

3.9 Προτεινόμενα Ενοικιαζόμενα (Recommended Rentals)

Τα **προτεινόμενα ενοικιαζόμενα (recommended rentals)** είναι μια οντότητα που δημιουργείται/ ανανεώνεται κατά τη λειτουργία του αλγόριθμου δημιουργίας εξατομικευμένης πρότασης. Περιέχει τον **ενοικιαστή (tenant)** και μια λίστα από προτεινόμενα **ενοικιαζόμενα (rentals).** Χρησιμοποιείται για τη δυνατότητα άμεσης άντλησης της πρότασης από τον τελικό χρήστη, χωρίς να χρειάζεται να υπολογιστεί η πρόταση σε real-time.

Κεφάλαιο 4 Σχεδιαστικές Επιλογές

4.1 Backend

Η υλοποίηση του κεντρικού server έχει χωριστεί στα παρακάτω packages:

- **domain**, που περιέχει τις οντότητες που αποθηκεύονται στη βάση δεδομένων.
- **repository,** που περιέχει διεπαφές(interfaces) για την αποθήκευση, άντληση, ανανέωση και διαγραφή οντοτήτων στη βάση (πιο συγκεκριμένα, διεπαφές που υλοποιούν το *<JpaRepository>* του Spring framework).
- **controller**, που περιέχει τις συναρτήσεις επικοινωνίας με τον εμπρόσθιο server,
- **dto**, που περιέχει τις δομές κανονικοποίησης της επικοινωνίας με τον εμπρόσθιο server
- **services**, που περιέχει την πιο λεπτομερή υλοποίηση της επεξεργασίας και του ελέγχου των εισερχόμενων και εξερχόμενων δεδομένων
- **configuration**, που περιέχει συναρτήσεις για τη ρύθμιση της ασφάλειας της επικοινωνίας με τον εμπρόσθιο server και τη διαχείριση του *cors filter,*
- **utils**, που περιέχει λειτουργίες απαραίτητες για τη δημιουργία και σωστή λειτουργία των παραγόμενων *jwt,* και
- **database**, που ομαδοποιεί τα δεδομένα της βάσης με κατάλληλο τρόπο, ώστε να έχουν συγκεκριμένη δομή πριν ληφθούν από τον διαχειριστή σε μορφή .json ή .xml

4.2 Frontend

Όλα τα είδη επικοινωνίας με τον κεντρικό server επιτυγχάνονται με τη χρήση του ***fetch api***. Ο εμπρόσθιος server ύστερα περιμένει την απάντηση και πράττει ανάλογα με τον *http κωδικό απάντησης* και τα επιστρεφόμενα δεδομένα (αν υπάρχουν), τα οποία επεξεργάζεται κατάλληλα πριν εμφανιστούν στον end-user.

Η βάση της ιεαρχίας αρχείων στο frontend είναι το αρχείο App.js, στο οποίο εισάγεται (import) κάθε άλλο αρχείο React που αντιστοιχεί σε κάποιο view του site και συνδέεται με ένα συγκεκριμένο route. Τα βασικά routes είναι :

- **/auth**: Views σχετικά με την **εγγραφή**, **σύνδεση** και **αποσύνδεση** χρήστη , καθώς και αντιμετώπιση μη εξουσιοδοτημένης αίτησης λόγω μη σύνδεσης του χρήστη.

- **/**: Views στα οποία έχουν πρόσβαση **εγγεγραμμένοι και ανώνυμοι** χρήστες. Στα συγκεκριμένα views, οι χρήστες κάνουν αναζήτηση δωματίων, τους δίνεται η δυνατότητα να τους προταθούν δωμάτια με βάση τις προτιμήσεις τους και αν είναι εγγεγραμμένοι, να δουν και να αλλάξουν το προφίλ τους. Για την αναζήτηση δωματίων, οι χρήστες αναγκαστικά πρέπει να εισάγωγουν ημερομηνίες και αριθμό ατόμων και προαιρετικά πληροφορίες για την τοποθεσία αναζήτησης. Όταν εισάγουν τα στοιχεία αναζήτησης, θα τους εμφανιστεί μία με τα διαθέσιμα δωμάτια σύμφωνα με τις πληροφορίες που δόθηκαν. Τα αποτελέσματα είναι σελιδοποιημένα και δίνεται η

επιλογή να μετακινηθεί ο χρήστης στην επόμενη, προηγούμενη, πρώτη και τελευταία σελίδα. Τέλος, δίνεται η δυνατότητα να προστεθούν επιπλέον φίλτρα. Τα δωμάτια της λίστας εμφανίζονται σε μορφή πλέγματος και είναι κουμπιά, τα οποία αν ο χρήστης πατήσει πλοηγείται στη σελίδα λεπτομερούς εμφάνισης του δωματίου. Εκεί έχει την επιλογή να δει τις πληροφορίες του δωματίου, να κάνει κράτηση, να προσθέσει κριτική και να στείλει μήνυμα στον ιδιοκτήτη, πατώντας πάνω στο πλέγμα που δείχνει τα στοιχεία του. Εμφανίζονται 18 μηνύματα ανά σελίδα ταξινομημένα με βάση τον χρόνο αποστολής σε φθίνουσα σειρά, και ο χρήστης έχει την επιλογή να πλοηγηθεί στα προηγούμενα και επόμενα μηνύματα μέσω κουμπιών αλλαγής σελίδας.

- **/admin**: Αποτελούν views στα οποία έχουν πρόσβαση μόνο οι **διαχειριστές**. Στα συγκεκριμένα views, χρησιμοποιούμε την **useEffect** συνάρτηση της React, η οποία εκτελείται κατά την πρώτη εμφάνιση του view στο οποίο ορίζεται και μετά από αυτό, κάθε φορά που αλλάζει η τιμή κάποιας μεταβλητής που έχει οριστεί ως εξάρτηση (dependency) της useEffect, στην οποία κάνουμε μία κλήση στο backend στέλνοντας το jwt, για να επιβεβαιώσουμε ότι ο χρήστης είναι διαχειριστής. Σε αυτό το route βρίσκονται οι επιλογές λήψης της βάσης σε .json ή .xml μορφή, καθώς και και να δει μία λίστα με όλους τους χρήστες, σελιδοποιημένη με παρόμοιο τρόπο που έχουν σελιδοποιηθεί τα δωμάτια μετά από μία αναζήτηση. Ομοίως με τα δωμάτια, κάθε πλέγμα χρήστη είναι ένα κουμπί που αν πατηθεί δείχνει τις λεπτομέρειες στοιχείων του χρήστη και δίνει την επιλογή ενεργοποίησής του, αν είναι οικοδεσπότης.

- **/host:** Αποτελούν views στα οποία έχουν πρόσβαση μόνο **οικοδεσπότες**. Εκεί, ο οικοδεσπότης μπορεί να δει τα δωμάτια που έχει βάλει για ενοικίαση, να δει το προφίλ του και να προσθέσει νέα δωμάτια για ενοικίαση. Επίσης, όταν πληγηθεί στις λεπτομέρειες ενός δωματίου, μπορεί να δει τη λίστα με τους χρήστες που του έχουν στείλει μήνυμα για το συγκεκριμένο δωμάτιο και να τους απαντήσει.

- **/tenant**: Αποτελούν views στα οποία έχουν πρόσβαση μόνο **ενοικιαστές**. Τα συγκεκριμένα views έχουν να κάνουν μόνο με τις κρατήσεις και τις κριτικές και υπάρχουν για το διαχωρισμό των ανώνυμων χρηστών και των συνδεδεμένων ενοικιαστών.

Κεφάλαιο 5 Λεπτομέρειες Υλοποίησης

5.1 CORS Filter

Για την διαχείριση του CORS, ώστε η επικοινωνία backend - frontend - browser να είναι εφικτή, στον φάκελο
{project_folder}/src/main/java/com/WebApplicatinTechnologies/AirBnBClone/configuration
μας ενδιαφέρουν τα εξής αρχεία :

Στο αρχείο SecurityConfiguration και πιο συγκεκριμένα στη συνάρτηση filterChain, στην ιεαρχία φιλτραρίσματος κάθε request για τον έλεγχο ασφάλειας, υπάρχει η εντολή ***.addFilterBefore(new CorsFilter(), ChannelProcessingFilter.class)*** , μέσω της οποίας γίνεται η πρώτη ανταλλαγή πληροφορίας του frontend με το backend για την μετάδοση των χρήσιμων CORS headers.

Στο αρχείο CorsFilter και πιο συγκεκριμένα στη συνάρτηση doFilter, η οποία χρησιμοποιείται απ' το παραπάνω φίλτρο, δέχεται το request απ΄το frontend και σε περίπτωση που η μέθοδος του request είναι OPTIONS, δηλαδή είναι CORS request,
δημιουργεί ένα HTTP Response, στο οποίο επισυνάπτει τα εξής headers :

"Access-Control-Allow-Origin" : "*" => Το συγκεκριμένο header επικοινωνεί στο frontend ότι δέχεται requests από κάθε domain. Σε κανονικές συνθήκες, αυτό το header θα έπρεπε να περιορίζεται μόνο στα domain που χρησιμοποιεί το frontend (πχ localhost:3000) και όχι * , που σημαίνει όλα τα domain. Καθώς στην συγκεκριμένη περίπτωση το web site βρίσκεται στα πλαίσια εργασίας, χρησιμοποιείται * για την διευκόλυνση της διόρθωσης.

"Access-Control-Allow-Methods" : "POST, GET, OPTIONS, DELETE, PUT" => Το συγκεκριμένο header επικοινωνεί στο frontend ότι δέχεται requests με της μεθόδους που προσδιορίζονται (δηλαδή, "POST, GET, OPTIONS, DELETE, PUT")

"Access-Control-Max-Age", "3600" => Το συγκεκριμένο header προσδιορίζει στο frontend τον χρόνο που θα αποθηκεύσει τις πληροφορίες που δέχθηκε απ΄ το HTTP Response στο CORS request.

"Access-Control-Allow-Headers" : "*" => Το συγκεκριμένο header επικοινωνεί στο frontend ότι δέχεται requests με headers που προσδιορίζονται (δηλαδή, όλα τα headers)

5.2 Ιστορικό Μηνυμάτων

Λόγω του ότι η σελιδοποίηση κατά την εμφάνιση του ιστορικού μηνυμάτων μεταξύ οικοδεσπότη και ενοικιαστή δεν βασίζεται σε πεδίο της δομής ***MessageHistory***, αλλά σε πεδίο ενός μέλους ενός πεδίου της (συγκεκριμένα MessageHistory::messageList::Message::sentAt) **δεν** γίνεται με τη χρήση του αντίστοιχου repository με χρήση της διεπαφής ***Pageable*** του ***Spring*** . Αντίθετα γίνεται με την άντληση του κατάλληλου ***MessageHistory***, ταξινόμησης της λίστας μηνυμάτων, και εμφάνισης της υπο-λίστας (*Java.util.List.subList()*) που αντιστοιχεί στον αιτούμενο αριθμό

σελίδας. Η σελιδοποίηση σε άλλες λειτουργίες, όπως η αναζήτηση ενοικιαζόμενων από ενοικιαστή ή χρηστών από διαχειριστή, γίνεται κανονικά με τη χρήση της αντίστοιχης δομής αποθήκευσης (repository).

5.3 Οντότητα User

Για τη διευκόλυνση -μεταξύ άλλων- της υλοποίησης του ***SecurityConfiguration***, και πιο συγκεκριμένα των παραμέτρων που επιτρέπουν πρόσβαση σε μια σελίδα με βάση τους ρόλους του χρήστη, η δομή ***User*** υλοποιεί τη διεπαφή (interface) ***UserDetails*** του ***Spring*** .

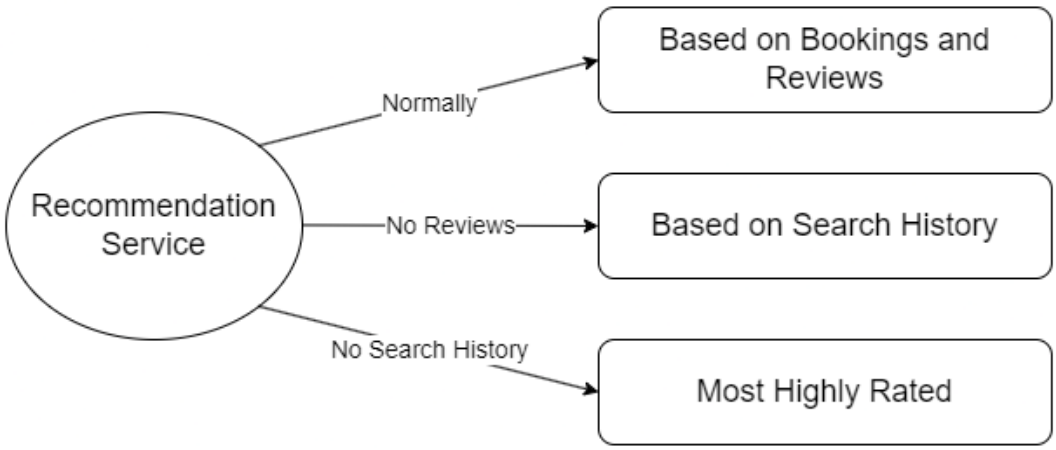
5.4 Οντότητα Role

Για τον ίδιο λόγο, η δομή ***Role*** υλοποιεί τη διεπαφή (interface) ***GrantedAuthority*** του ***Spring***.

5.5 Αλγόριθμος Εξατομικευμένης Πρότασης Ενοικιαζόμενων

Βρίσκεται στο *{project_folder}/src/main/java/com/WebAppTechnologies/AirBnBClone/services/RecommendationService.java* και υλοποιείται σε τρία στάδια:

1) recommendMostHighlyRated()
2) recommendBasedOnBookingsAndReviews()
3) recommendBasedOnSearchHistory()



5.5.1 recommendMostHighlyRated

Ο απλούστερος αλγόριθμος, και αυτός που χρησιμοποιείται όταν δεν υπάρχουν αρκετά δεδομένα για τη χρήση κάποιου από τους άλλους δύο. Πιο συγκεκριμένα, αυτός χρησιμοποιείται όταν η αίτηση για προτεινόμενα γίνεται από κάποιον που δεν έχει κάνει ούτε αναζητήσεις ούτε κρατήσεις στην εφαρμογή. Όπως προϊδεάζει και το όνομα, επιστρέφει τα ενοικιαζόμενα με τον καλύτερο μέσο όρο κριτικών (σε αστέρια), εφ' όσον βέβαια πληρούν έναν ελάχιστο αριθμό κριτικών.

Οι παρακάτω αλγόριθμοι χρησιμοποιούν τη μέθοδο της **Κατηγοριοποίησης Πινάκων με Κάθοδο Κλίσης** **(Matrix Factorization with Gradient Descent)** για τον υπολογισμό της εκτιμώμενης κριτικής ενός ενοικιαζόμενου από έναν χρήστη (***Ratings***) μέσω της άντλησης χαρακτηριστικών για τα ενοικιαζόμενα (***RentalFeatures***) και την εκτίμηση σημαντικότητας του κάθε χαρακτηριστικού για κάθε χρήστη (***UserFeatures***). *Σημείωση*: Για τον υπολογισμό των πινάκων δεν χρησιμοποιούνται δεδομένα από όλους τους χρήστες, αλλά από όσους έχουν υποβάλλει κριτική, και η τιμή του αρχικού πίνακα είναι ίση με τις κριτικές των χρηστών, αν υπάρχουν, και μια δεδομένη τιμή, αλλιώς. Για την περαιτέρω προσομοίωση των κριτικών, οι τιμές των κελιών των πινάκων είναι δεκαδικοί μεταξύ 1 και 5. Επίσης, για την αντιστοίχιση κάθε εκτιμώμενης κριτικής με ενοικιαζόμενο, χρησιμοποιείται η εσωτερική δομή (RentalInfo) που αποτελείται από ένα αντίγραφο του ενοικιαζόμενου, και την εκτιμώμενη κριτική του χρήστη.

5.5.2 recommendBasedOnBookingsAndReviews

Ο αλγόριθμος που χρησιμοποιείται πάντα όταν είναι εφικτό. Η εκτιμώμενη κριτική του εν λόγω χρήστη για ένα ενοικιαζόμενο είναι ίση με την τελευταία χρονολογικά κριτική του χρήστη, αν υπάρχει, αλλιώς είναι ανάλογη του αριθμού των φορών που έχει κάνει κράτηση με το συγκεκριμένο ενοικιαζόμενο.

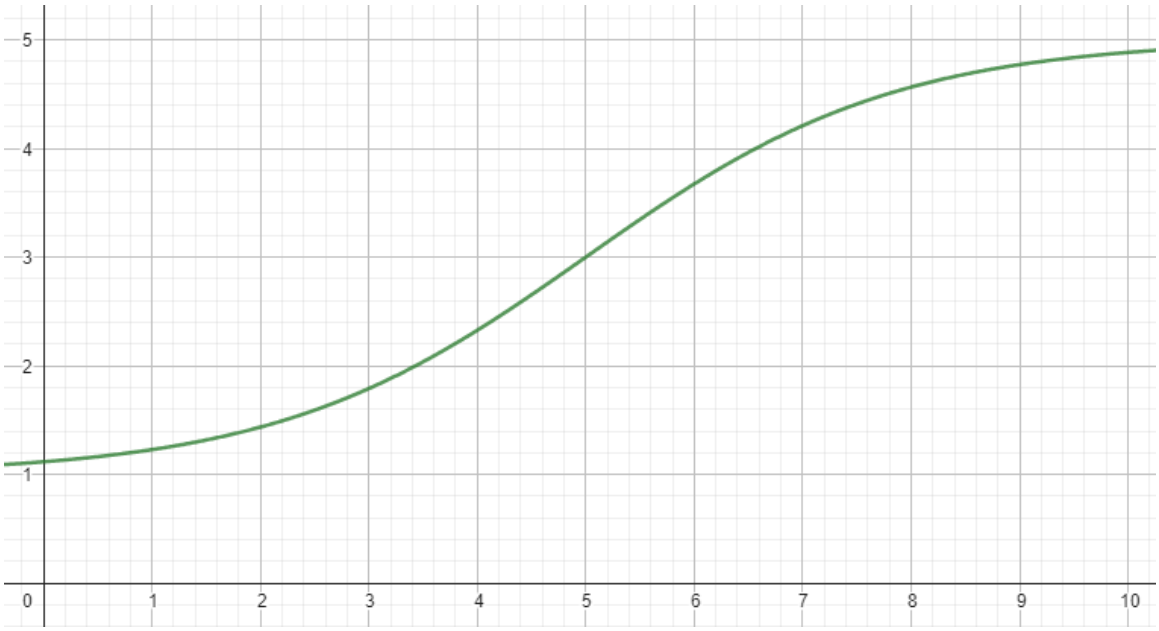
5.5.3 recommendBasedOnSearchHistory

Όπως υποδεικνύει το όνομα της συνάρτησης, αυτός ο αλγόριθμος χρησιμοποιείται για να προτείνει ενοικιαζόμενα με βάση το ιστορικό αναζήτησης του χρήστη. Αυτό περιλαμβάνει

a. αναζητήσεις με βάση πληροροφίες περιοχής /αριθμό ατόμων,
b. ενοικιαζόμενα των οποίων τη σελίδα έχει επισκεφθεί, καθώς και ο αριθμός των φορών που επισκέφθηκε τη σελίδα, και
c. τα φίλτρα που έχει επιλέξει σχετικά με τις παροχές του ενοικιαζόμενου.

Αρχικά υπολογίζονται οι βαρύτητες / σημαντικότητες των μεταβλητών (πχ αν στο 90% των αναζητήσεών του, ο χρήστης έχει πατήσει το checkbox “A/C”, τότε είναι σημαντικό για εκείνον το ενοικιαζόμενο να έχει air condition κ.ο.κ)

Ύστερα υπολογίζεται ένα «σκορ ομοιότητας» του κάθε ενοικιαζόμενου με τις αναζητήσεις του χρήστη, και το αποτέλεσμα ορίζεται από μια ***σιγμοειδή συνάρτηση*** με αποτέλεσμα μεταξύ 1 και 5. Ο λόγος που επιλέχθηκε σιγμοειδής συνάρτηση, είναι για την αξιοποίηση όλου του εύρους [1,5] των κριτικών. Καθώς σχεδόν όλα τα ενοικιαζόμενα θα έχουν μια στοιχειώδη ομοιότητα με το ιστορικό αναζήτησης, και πολύ λίγα θα έχουν πολύ υψηλή, κανένα ενοικιαζόμενο δεν θα είχε πολύ υψηλή ή πολύ χαμηλή κριτική. Η συνάρτηση λειτουργεί συνεπώς σαν μια μέθοδος κανονικοποίησης των δεδομένων.



Η διαδικασία άντλησης προτεινόμενων ενοικιαζόμενων εκτελείται αυτόματα κάθε 24 ώρες, και οι μεταβλητές της (ρυθμός εκμάθησης, αριθμός χαρακτηριστικών, αριθμός επαναλήψεων κ.ά) έχουν βελτιστοποιηθεί μετά από σχετικούς ελέγχους για ακρίβεια και ταχύτητα.

Κεφάλαιο 6 Επίλογος

Η εργασία ξεκίνησε με τη δημιουργία κάποιων βασικών δομών στο backend, όπως *User* και *Role*, και μια αρχική εξοικείωση με βασικές δομές του ***Spring*** framework όπως είναι τα *repositories, controllers, beans,* μεταξύ άλλων.

Ακολούθησε η ρύθμιση της ταυτοποίησης με τη χρήση jwt, που περιλαμβάνει τη δημιουργία, την κωδικοποίηση, την αποκωδικοποίηση, την αποστολή εκτός του server κλπ. Η αρχική υλοποίηση δεν λειτουργούσε, καθώς παρατηρήσαμε πως, όταν ένας χρήστης συνδεόταν στην εφαρμογή, το jwt που δημιουργούταν φαινομενικά δεν επέστρεφε. Όχι μόνο αυτό, ολόκληρη η κεφαλίδα (header) "Authorization" στην οποία ήταν αποθηκευμένο το jwt δεν έφτανε ποτέ στον χρήστη. Ύστερα καταλάβαμε πως η κεφαλίδα έφτανε, απλά δεν είχε πρόσβαση ο χρήστης. Συνεπώς, συνειδητοποιήσαμε πως έπρεπε να δημιουργήσουμε νέα κεφαλίδα με όνομα "Access-Control-Expose-Headers" με την τιμή "Authorization", και λειτουργούσε. Καταλήξαμε πως, αντί για δύο κεφαλίδες, ήταν προτιμότερο το jwt να είναι στο σώμα (body) της απάντησης. Ο χρήστης μετά το χρησιμοποιεί κανονικά στην κεφαλίδα "Authorization" στα επόμενα αιτήματά του.

Μέχρι εκείνο το σημείο, η αποστολή αιτημάτων στον server γινόταν μέσω της εφαρμογής ***Postman***, και όχι αποκλειστικού frontend server. Οπότε δημιουργήσαμε εφαρμογή react μέσω ***create-react-app***, και εξοικειωθήκαμε με μερικές βασικές λειτουργίες της react, όπως οι *useState, useEffect, localStorage* κ.ά.

Ακολούθησε η κατανόηση της ασύγχρονης φύσης του ***fetch api*** , η επακόλουθη χρησιμότητα της *.then(),* και η ανάγκη επεξεργασίας των δεδομένων που στέλνει, και δέχεται ο react server μέσω εντολών όπως *JSON.stringify()* και *{response}.json().*

Ήρθαμε αντιμέτωποι με απαγόρευση επικοινωνίας των δύο server λόγω ***CORS***, και συγκεκριμένα του ***"pre-flight request"*** και, αφότου η πρόσθεση "Cross-Origin-…" headers και η επισημείωση των controllers με "@CrossOrigin(*)" δεν λειτούργησε, καταλήξαμε πως ήταν αναγκαία η δημιουργία ειδικού φίλτρου για την αντιμετώπιση του προβλήματος.

Τέλος, στο θέμα του ***recommendation***, η αρχική πρόκληση ήταν πως δεν βρήκαμε εργαλεία της Java που να βοηθούν στην υλοποίηση του αλγορίθμου, οπότε πολλές λειτουργίες, όπως ο πολλαπλασιασμός δύο πινάκων ***(dotProduct)***, ή η άντληση μιας στήλης από έναν δισδιάστατο πίνακα ***(getColumn)*** έπρεπε να γραφτούν από την αρχή. Η συνάρτηση όμως ήταν και πάλι πολύ μεγάλη και δυσνόητη, οπότε μια πρόταση ήταν η χρήση του αντικειμενοστραφούς χαρακτήρα της γλώσσας, ορίζοντας προσωρινά τα *UserFeatures* και *RentalFeatures* ως πεδία της δομής. Τελική μορφή είναι μια μέση λύση, με στοιχεία αντικειμενοστραφούς χαρακτήρα (οι μεταβλητές είναι στατικές, οι τιμές των πινάκων είναι στιγμιότυπα κλάσης) χωρίς περαιτέρω προσπάθεια χρήσης των εργαλείων της Java, καθώς πολύ πιθανόν να μείωναν και τη χρονική απόδοση.